Further Reducing the Redundancy of a Notation Over a Minimally Redundant Digit Set

Marc Daumas, David Matula

Journal of VLSI Signal Processing 33, 7–18, 2003

Further Reducing the Redundancy of a Notation Over a Minimally Redundant Digit Set

MARC DAUMAS*
Laboratoire de l'Informatique du Parallélisme, CNRS, ENS de Lyon, France

DAVID W. MATULA
Southern Methodist University, School of Engineering, Dallas, Texas, USA

Received September 25, 2001; Revised November 30, 2001

Abstract. Redundant notations are used implicitly or explicitly in many digital designs. They have been studied in details and a general framework is known to reduce the redundancy of a notation down to the minimally redundant digit set. We present here an operator to further reduce the redundancy of such a representation. It does not reduce the number of allowed digits since removing one digit to a minimally redundant digit set is a conversion to a non redundant digit set and this is an expensive operation. Our operator introduces some correlation between the digits to reduce the number of possible redundant notations for any represented number. This reduction is visible in small useful operators like the elimination of leading zeros. We also present a key application with a CMOS Booth recoded multiplier. Our multiplier is able to accept both a redundant or a non redundant input with very little modifications and almost no penalty in time or space compared to state-of-the-art non redundant multipliers.

Keywords: computer arithmetic, redundant notation, addition, multiplication

1. Introduction and Summary

Any positive integer has a unique radix β representation ($\beta \in \mathbb{N}$ and $\beta \geq 2$) with digits in the set $\{0, \ldots, \beta - 1\}$. When the set contains more than β elements, the notation is redundant and many integers have several representations [1–4]. The redundancy increases with the number of additional elements in the digit set.

In some cases, we are able to retrieve some correlations between the digits of the redundant representation computed by an algorithm. That assertion reduces the redundancy of the notation even though the digit set is not changed.

One such example is the radix 4 recoding inspired from Booth [5] as presented in Koren [6]. The number is written on the digit set $\{-2, \ldots, 2\}$ but a digit 2 can only be followed by a negative digit possibly preceded by a string of zeros. For example, $(201)_4 = 2 \cdot 4^2 + 0 \cdot 4^1 + 1 \cdot 4^0 = 33$ is not a valid Booth recoded representation. The valid one for 33 is $(1\bar{2}01)_4 = 1 \cdot 4^3 + (-2) \cdot 4^2 + 0 \cdot 4^1 + 1 \cdot 4^0$.

The conversion from the usual radix 2 representation to the radix 4 Booth recoded representation is obtained from the bit to digit conversion presented in Table 3. This special radix 4 notation on the redundant digit set $\{-2, \ldots, 2\}$ is not redundant.

In Section 2, we identify a quantity that describes part of the correlation of the digits. We have first presented this quantity in [7], it has been used by Matula and Nielsen [8], Nielsen et al. [9], Seidel and Even [10], Dumonteix and Mehrez [11], and refered by Seidel [12]. It is called the fraction range associated to a set of valid representations (language). It is the set of all the possible fractions of units in the last position lost when

*This work has been partially supported by the PICS 479 from the French National Center for Scientific Research (CNRS).

one truncates any valid representation for any possible position in the representation.

The fraction range of a general radix 4 notation on the digit set $\{-2, \ldots, 2\}$ is included in $(-2/3, 2/3)$ whereas the fraction range of the abovementioned Booth recoded notation is included in $[-1/2, 1/2]$. We will see a first example of how this quantity measures the redundancy of the notation in the Section 2.3 where the window of k digits of a radix β representation forms a new representation radix β^k.

Fraction range proves to be appropriate in retaining most of the information when carry recodings are applied. Such operations are specific digit set conversions [13] or rewriting rules [14, 15] that compute in parallel a ± 1 carry and an in-place residual digit for each digit position. Digit set conversion is a generalization of the addition of numbers in redundant and/or non redundant notations on parallel and serial systems. For example, the binary sum of $(101101)_2$ and $(011101)_2$ can be implemented as the radix 2 conversion of $(112202)_2$ to the digit set $\{0, 1\}$. The fraction range traces the correlation of the digits in a complex arithmetic algorithm that uses only recodings as this is the case for most implemented arithmetic operators on a redundant notation.

We present in Section 3 three applications of redundancy reduction of radix two borrow save and carry save notations, one of them being our key Booth multiplier with one redundant operand [16].

2. Fraction Range and Carry Recoding

2.1. Definitions

The radix β notation $(d_m \ldots d_0 \cdot d_{-1} \ldots d_l)_\beta$ on the digit set S represents the rational number of Eq. (1). The set of all the possible representations from positions l through m with digits in S is the formal language noted S_l^m $(l \le 0 \le m)$. The valuation $\|\cdot\|_\beta$ maps S_l^m into the set Q of the rational numbers.

$$\|d_m \ldots d_0 \cdot d_{-1} \ldots d_l\|_\beta = \sum_{i=l}^{m} d_i \beta^i \quad \text{with} \quad l \le i \le m \Rightarrow d_i \in S \tag{1}$$

In the following, we are only interested in contiguous digits sets containing zero, that is $S = \{S_{\min}, \ldots, S_{\max}\}$ with $S_{\min} \le 0 \le S_{\max}$ leading to a redundant notation as soon as $S_{\max} - S_{\min} \ge \beta$. The digit set is minimally redundant if it contains exactly $\beta + 1$ digits, that is $S_{\max} - S_{\min} = \beta$. A carry ripple process with best time complexity $\Theta(\log(m - l))$ is necessary to convert to a notation with only β elements.

On a computer, a number is always produced as a vector of digits $D = [d_m \ldots d_0\ d_{-1} \ldots d_l]$. An algorithm that computes $A(X) = D$ does not produce a rational number (the valuation) but a vector of digits (the representation). In the introduction, we have seen that there might be some correlations between the digits of the vector as long as they have not been given by the user but computed. Without knowing the actual value of the digits stored in the vector D, we can define some representations that are acceptable and prove that some others cannot occur. This defines the language $A^+ = \{A(X) \text{ where } X \text{ is a possible input}\} \subset S_l^m$ which is the subset of all the possible output representations. By extension, we use the same notation for A^+ and A in the following equations. We define the fraction range of A and the fraction range at position j by Eqs. (2) and (3). We will see how this single set captures alone some very relevant part of the correlation between the digits.

$$\begin{aligned} Fr_j(A) &= Fr_j(A^+) \\ &= \{\|0 \cdot d_{j-1} \ldots d_l\|_\beta, [d_m \ldots d_l] \in A^+\} \quad (2) \\ Fr(A) &= Fr(A^+) = \bigcup_{l \le j \le m} Fr_j(A^+) \quad (3) \end{aligned}$$

For a vector that is input by the user digit after digit, nothing is known about any correlation. This yields the largest a priori fraction range associated with the given radix and the given digit set. It is bounded by Eq. (4).[1]

$$\begin{aligned} Fr\left(S_l^m\right) &= \left\{\|0 \cdot d_{j-1} \ldots d_l\|_\beta, l \le j \le m, \right. \\ &\qquad \left. [d_m \ldots d_l] \in S_l^m\right\} \\ Fr\left(S_l^m\right) &\subset \frac{1}{\beta - 1} \cdot (S_{\min}; S_{\max}) \end{aligned} \tag{4}$$

For radix two representations, the fraction range of a borrow save vector where $S = \{-1, 0, 1\}$ and a carry save vector where $S = \{0, 1, 2\}$ are respectively $(-1, 1)$ and $[0, 2)$. Indeed this fact was recognized in Intel's description of the Pentium bug: truncating a number in carry save format produces an error within 2 ulps called the "region of uncertainty" [17, 18]. If the number is stored in borrow save format the error is within ± 1 ulp.

A positive carry recoding transforms $(d_m \ldots d_l)_\beta$ on the digit set $S = \{S_{\min}, \ldots, S_{\max}\}$ to the representation $(e_{m+1}\ e_m \ldots e_l)_\beta$. It uses a quantity $p \in S$ called the pivot to define the Eqs. (5) and (6) for $l \le i \le m + 1$

where $c_l = d_{m+1} = c_{m+2} = 0$.

$$c_{i+1} = \begin{cases} 1 & \text{if } d_i \geq p \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$c_{i+1}\beta + e_i = d_i + c_i \quad (6)$$

We verify by induction on m that a recoding does not change the value stored as presented below.

$$\sum_{i=l}^{m} d_i \beta^i = \sum_{i=l}^{m+1} e_i \beta^i$$

The recoded digit set is $\{\min(S_{\min}; p - \beta), \ldots, \max(S_{\max} - \beta + 1; p)\}$. The digit set is reduced if $S_{\min} + \beta \leq p < S_{\max}$. The output digit set is minimally redundant as soon as $S_{\min} \geq p - \beta$ and $p + \beta > S_{\max}$, that is the case for example if the digit set is maximally redundant and $p = 0$. If the output digit set is minimally redundant, it is $\{p-\beta, \ldots, p\}$. If $p = S_{\max} = S_{\min} + \beta$ the output digit set is minimally redundant and identical to the input set.

A negative carry conversion is similar to the positive carry conversion except that c_{i+1} is defined from Eq. (7).

$$c_{i+1} = \begin{cases} -1 & \text{if } d_i \leq p \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

The recoded digit set is $\{\min(S_{\min} + \beta - 1; p), \ldots, \max(S_{\max}; p + \beta)\}$. The digit set is reduced if $S_{\min} < p \leq S_{\max} - \beta$. If the output digit set is minimally redundant, it is $\{p, \ldots, p + \beta\}$. It is unchanged, minimally redundant, if $p = S_{\min} = S_{\max} - \beta$. We will see that even when we do not reduce the number of elements of the digit set, such conversions perform some legitimate work visible through its effect on the fraction range.

2.2. *Fundamental Property*

If A is a language of S_l^m with fraction range $Fr(A)$ and P any positive carry recoding with output digit set $\{S_{\min}, \ldots, S_{\max}\}$ then[2]

$$Fr(P(A)) \subset \frac{1}{\beta}(Fr(A) + \{S_{\min}, \ldots, S_{\max} - 1\}). \quad (8)$$

Proof: Let $D \in A$ with $D = [d_m \ldots d_l]$. We define the carry vector $C = [c_{m+1}\ c_m \ldots c_{l+1}]$ from Eq. (5) and the result vector $E = P(D) = [e_{m+1}\ e_m \ldots e_l]$ from Eq. (6). We obtain the following equation for any rational number $f \in Fr_j(P(A))$ by extracting a term from the radix polynomial of E in Eq. (1)

$$\begin{aligned} f &= \|0 \cdot e_{j-1} \ldots e_l\|_\beta \\ &= \frac{e_{j-1} - c_{j-1}}{\beta} + \|0 \cdot c_{j-1}\ e_{j-2} \ldots e_l\|_\beta. \end{aligned}$$

We recognize that $[c_{j-1}\ e_{j-2} \ldots e_l]$ is the recoded vector of $[d_{j-2} \ldots d_l]$ and therefore

$$\begin{aligned} \|0 \cdot c_{j-1}\ e_{j-2} \ldots e_l\|_\beta &= \|0 \cdot 0\, d_{j-2} \ldots d_l\|_\beta \\ &\in \frac{1}{\beta} Fr_{j+1}(A) \subset \frac{1}{\beta} Fr(A). \end{aligned}$$

We know from the definition of the output digit set that $e_{j-1} - c_{j-1} = d_{j-1} - \beta c_j < S_{\max}$ to allow c_{j-1} to be incorporated without carry ripple. It follows that

$$\begin{aligned} Fr(P(A)) &\subset \bigcup_{k=S_{\min}}^{S_{\max}-1} \frac{1}{\beta}(\{k\} + Fr(A)) \\ &\subset \frac{1}{\beta}(Fr(A) + \{S_{\min}, \ldots, S_{\max} - 1\}). \end{aligned}$$

□

We prove in a very similar way that if N is a negative carry recoding then

$$Fr(N(A)) \subset \frac{1}{\beta}(Fr(A) + \{S_{\min} + 1, \ldots, S_{\max}\}). \quad (9)$$

Considering Eqs. (8) and (9) it is now legitimate to perform a digit set recoding from one digit set to the same digit set since it reduces the fraction range of the final vector. We can recode a borrow save vector to reduce its fraction range from (a, b) to $(-1/2 + a/2, b/2)$ with a positive recoding and to $(a/2, b/2 + 1/2)$ with a negative recoding. For a carry save number, we can reduce its fraction range from $[a, b)$ to $[a/2, b/2 + 1/2)$ with a positive recoding and to $[1/2 + a/2, b/2 + 1)$ with a negative recoding.

We start with digits in the set $\{p - \beta, \ldots, p - 1 + r\}$ with $1 \leq r \leq \beta$ for a generalization of this last observation. Let P be a positive recoding with pivot p, then by induction

$$Fr(P^k) \subset \frac{1}{\beta - 1} \cdot \left(p - \beta, p - 1 + \frac{r}{\beta^k}\right) \cup \{0\}. \quad (10)$$

Comparing the fraction range of such a vector recoded k times to the fraction range of a non redundant

vector $(\beta-1)^{-1}\cdot(p-\beta;p-1)$, we see that the difference $r\cdot\beta^{-k}$ is decreasing geometrically with each new recoding. We can even apply another recoding to center the added quantity. Let N be the negative recoding with pivot $p-\beta$, the fraction range $NP^k(X)$ is bounded by Eq. (11). The centering is best if $\beta=2$ since $(\beta-1)/\beta=1/\beta=1/2$.

$$Fr(NP^k) \subset \frac{1}{\beta-1}\cdot\left(p-\beta+\frac{\beta-1}{\beta};\ p-1+\frac{\beta-1}{\beta}+\frac{r}{\beta^{k+1}}\right)\cup\{0\} \quad (11)$$

Proof: The proof of Eq. (10) is easily obtained by induction on k. We prove it here just for $k=1$. From Eq. (4), we know that $Fr(S_l^m)\subset(\beta-1)^{-1}\cdot(p-\beta,\ p-1+r)\cup\{0\}$. The output digit set is minimally redundant since $p-1+r<p+\beta$ and Eq. (8) gives us the inclusion:

$$\begin{aligned}Fr(P) &\subset \frac{1}{\beta}\left(Fr\left(S_l^m\right)+\{p-\beta,\ldots,p-1\}\right)\\ &\subset \frac{1}{\beta}\left(\frac{1}{\beta-1}(p-\beta)+p-\beta;\ \frac{1}{\beta-1}(p-1+r)+p-1\right)\cup\{0\}\\ &\subset \frac{1}{\beta-1}\left(p-\beta;p-1+\frac{r}{\beta}\right)\cup\{0\}.\end{aligned}$$

Equation (11) is proved in a similar way:

$$\begin{aligned}Fr(NP^k) &\subset \frac{1}{\beta}(Fr(P^k)+\{p-\beta+1,\ldots,p\})\\ &\subset \frac{1}{\beta}\left(\frac{1}{\beta-1}(p-\beta)+p-\beta+1;\ \frac{1}{\beta-1}\left(p-1+\frac{r}{\beta^k}\right)+p\right)\cup\{0\}\\ &\subset \frac{1}{\beta-1}\left(p-\beta+\frac{\beta-1}{\beta};p-1+\frac{\beta-1}{\beta}+\frac{r}{\beta^{k+1}}\right)\cup\{0\}.\end{aligned}$$

□

We obtain an identical property if k negative recodings are applied to a vector of digits in the set $\{p+1-r,\ldots,p+\beta\}$ with $1\le r\le\beta$. In this case N^k has its fraction range bounded by Eq. (12) and PN^k with one last positive recoding of pivot $p+\beta$ has its fraction range bounded by Eq. (13).

$$Fr(N^k)\subset\frac{1}{\beta-1}\cdot\left(p+1-\frac{r}{\beta^k},\ p+\beta\right)\cup\{0\} \quad (12)$$

$$Fr(PN^k)\subset\frac{1}{\beta-1}\cdot\left(p+1-\frac{\beta-1}{\beta}-\frac{r}{\beta^{k+1}};\ p+\beta-\frac{\beta-1}{\beta}\right)\cup\{0\} \quad (13)$$

2.3. *First Application: Combining Several Digits in a Window*

We will see in Section 3 how the fraction range of a binary encoding can be used to deduce important properties on the digits stored in a vector. We will just present here a first practical application of the fraction range.

Let $D=[d_m\ldots d_l]$ be a vector of radix β digits. The window of k digits starting at position $j\cdot k$ can be valued alone as an integer d'_j given by Eq. (14) with digits in the range of Eq. (15). If we read the digits d'_j for any $\lfloor m/k\rfloor\le j\le\lfloor l/k\rfloor$ we obtain a new vector representing the same number radix β^k as shown in Eq. (16). We commonly use the octal and the hexadecimal notations because this conversion is very easy to perform back and forth with $\beta=2$ and $k=3$ or $k=4$.

$$d'_j=\left\|d_{(j+1)\cdot k-1}\ldots d_{j\cdot k}\right\|_\beta=\sum_{i=0}^{k-1}d_{j\cdot k+i}\beta^i \quad (14)$$

$$d'_j\in\frac{\beta^k-1}{\beta-1}\cdot[S_{\min},S_{\max}] \quad (15)$$

$$\|d_m\ldots d_l\|_\beta=\left\|d'_{\lfloor\frac{m}{k}\rfloor}\ldots d'_{\lfloor\frac{l}{k}\rfloor}\right\|_{\beta^k} \quad (16)$$

We can also write d'_j as presented in Eq. (17) leading to another interval for d'_j as shown Eq. (18).

$$d'_j=\left\|0\cdot d_{(j+1)\cdot k-1}\ldots d_l\right\|_\beta\cdot\beta^k-\left\|0\cdot d_{j\cdot k}\ldots d_l\right\|_\beta \quad (17)$$

$$d'_j\in(\beta^k\cdot Fr(D)-Fr(D)) \quad (18)$$

If the fraction range satisfies $Fr(D)\subset(a,b)$, Eq. (18) yields that a combined digit d'_j is bounded by $a\cdot\beta^k-b<d'_j<b\cdot\beta^k-a$. We present Tables 1 and 2 two examples radix 10, with the digit set $\{-1,\ldots,9\}$ (generalized borrow save) and up to 3 positive carry

Table 1. Reducing the redundancy to obtain a minimally redundant digit set radix 100.

Rec.	Fraction range (radix 10)	Digits radix 100	
		Max	Digit set
Non	$\frac{1}{9}\cdot(-1;8+1)=\left(-\frac{1}{9};1\right)$	99	$\{-11,\ldots,99\}$
P	$\frac{1}{9}\cdot\left(-1;8+\frac{1}{10}\right)=\left(-\frac{1}{9};\frac{9}{10}\right)$	$90+\frac{1}{9}$	$\{-11,\ldots,90\}$
P^2	$\frac{1}{9}\cdot\left(-1;8+\frac{1}{100}\right)=\left(-\frac{1}{9};\frac{89}{100}\right)$	$89+\frac{1}{9}$	$\{-11,\ldots,89\}$[a]
P^3	$\frac{1}{9}\cdot\left(-1;8+\frac{1}{1000}\right)=\left(-\frac{1}{9};\frac{889}{1000}\right)$	$88.9+\frac{1}{9}$	$\{-11,\ldots,89\}$[a]

[a]The digit set is minimally redundant. There will be no further improvement.

Table 2. Reducing the redundancy to obtain a minimally redundant digit set radix 1000.

Rec.	Fraction range (radix 10)	Digits radix 1000	
		Max	Digit set
Non	$\frac{1}{9}\cdot(-1;8+1)=\left(-\frac{1}{9};1\right)$	999	$\{-111,\ldots,999\}$
P	$\frac{1}{9}\cdot\left(-1;8+\frac{1}{10}\right)=\left(-\frac{1}{9};\frac{9}{10}\right)$	$900+\frac{1}{9}$	$\{-111,\ldots,900\}$
P^2	$\frac{1}{9}\cdot\left(-1;8+\frac{1}{100}\right)=\left(-\frac{1}{9};\frac{89}{100}\right)$	$890+\frac{1}{9}$	$\{-111,\ldots,890\}$
P^3	$\frac{1}{9}\cdot\left(-1;8+\frac{1}{1000}\right)=\left(-\frac{1}{9};\frac{889}{1000}\right)$	$889+\frac{1}{9}$	$\{-111,\ldots,889\}$[a]

[a]The digit set is minimally redundant. There will be no further improvement.

recodings with pivot $p = 9$. The lower bound on the combined digit set is given by Eq. (15) and does not change. The maximum value for one digit is computed from the fraction range in the table. The first recoding removed approximately a portion $\frac{1}{\beta}$ of the possible digits and the second one a portion $\frac{1}{\beta^2}$.

Getting rid of almost $\frac{1}{\beta}$ of the digits allows us to reach easily the minimally redundant digit set. It may not seem to be important working radix 10 but this is very relevant radix 2 as we will see in the next section.

3. Binary Applications

As we have seen in Eqs. (10)–(13), a short sequence of recodings can be employed to reduce the fraction range and approach the width of 1 ulp. The practical value of binary carry recodings is that just a few recodings provide partial compression sufficient to obtain almost all the benefits of full conversion to a non redundant notation while avoiding the high cost of a carry-ripple addition. The following observations are straightforward from the definition and support partial compression applications in rounding, leading insignificant digit deletion, and multiplier recoding.

3.1. Implementation and Bit Specialization

In computers, the digit vector is not stored but encoded into bits. In the carry save format, each digit d_i is stored with two bits p_i and q_i and the digit value is defined as $d_i = p_i + q_i$. We specialize the bits of the result register $e_i = p'_i + q'_i$ such that p'_{i+1} stores the carry bit c_{i+1} generated by Eq. (5) and q'_i stores the residual quantity $e_i - c_i = d_i - c_{i+1}\beta$ of Eq. (6). This leads us to the truth table and the equations for the positive carry recoding of a carry save number Fig. 1(a). If the numbers are stored using two's complement, the sign digit is moved to position $m + 1$ and $q'_{m+1} = q'_m$. The equations of Fig. 1(a) are the ones of an half adder cell (HA) as we will see soon.

The negative carry recoding of a carry save number cannot be defined in such an elementary manner but we may define more recodings if the target encoding is borrow save instead of carry save. A borrow save digit d_i (resp. e_i) is stored with two bits p_i and n_i (resp. p'_i and n'_i) and the digit is defined as $d_i = p_i - n_i$ (resp. $e_i = p'_i - n'_i$). We specialize a positive carry or a negative carry recoding since we can store the carry in p'_{i+1} (positive carry P-recoding of Fig. 1(b) or in n'_{i+1} (negative carry N-recoding of Fig. 1(c)).

The negative carry recoding of a carry save number to borrow save notation is not possible but a positive

CS	p_i	0	0	1	1
in	q_i	0	1	0	1
Digit d_i		0	1	1	2
CS	p'_{i+1}	0	0	0	1
out	q'_i	0	1	1	0

Pivot $p = 2$
Carry $p'_{i+1} = p_i \cdot q_i$
Residual $q'_i = p_i \oplus q_i$

(a) Positive carry recoding of a carry save number (HA)

BS	p_i	0	0	1	1
in	n_i	0	1	0	1
Digit d_i		0	-1	1	0
BS	p'_{i+1}	0	0	1	0
out	n'_i	0	1	1	0

Pivot $p = 1$
Carry $p'_{i+1} = p_i \cdot \overline{n}_i$
Residual $n'_i = p_i \oplus n_i$

(b) Positive carry recoding of a borrow save number (P)

BS	p_i	0	0	1	1
in	n_i	0	1	0	1
Digit d_i		0	-1	1	0
BS	p'_i	0	1	1	0
out	n'_{i+1}	0	1	0	0

Pivot $p = -1$
Carry $n'_{i+1} = \overline{p}_i \cdot n_i$
Residual $p'_i = p_i \oplus n_i$

(c) Negative carry recoding of a borrow save number (N)

CS	p_i	0	0	1	1
in	q_i	0	1	0	1
Digit d_i		0	1	1	2
BS	p'_{i+1}	0	1	1	1
out	n'_i	0	1	1	0

Pivot $p = 1$
Carry $p'_{i+1} = p_i + q_i$
Residual $n'_i = p_i \oplus q_i$

(d) Positive carry recoding from carry save number to borrow save representation (Q)

Figure 1. Truth table and equations of binary recodings with a specialized carry bit.

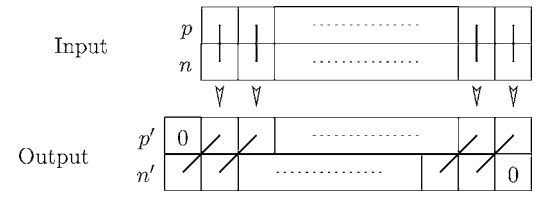


Figure 2. N-carry recoding of a borrow save number.

carry recoding is possible. It is presented Fig. 1(d) as the Q-recoding. Figure 2 illustrates how each output diagonal of the $2 \times (m-l+2)$ bit array is determined by an input column of the $2 \times (m-l+1)$ bit array for the N-carry recoding.

The half-adder cell is one of the basic cells of the computer arithmetic libraries. Functionally, it is implemented with an `exclusive or` gate and a `logical and` (see Fig. 3). In choosing the place to add some new `logical inverters`, we define the bit level cells corresponding to the P-, N- and Q-recodings (see Fig. 4). These inverters may or may not yield an actual penalty compare to a standard half adder cell. For example, the `exclusive or` gate may be implemented using pass transistors as it is the case in state-of-the-art circuit design [19, 20]. If so, an inverted `exclusive or` gate is obtained by switching some of the input wires of a straight `exclusive or` gate.

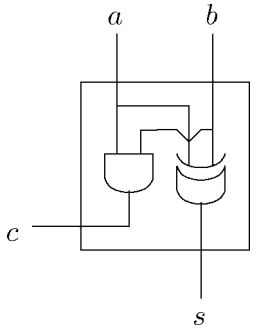


Figure 3. Half adder cell (HA) $a + b = 2c + s$.

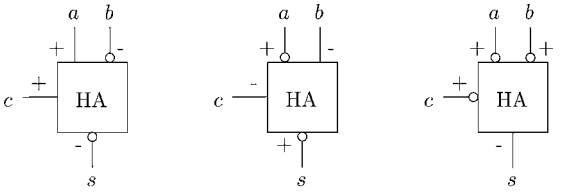


Figure 4. Recoder cells.

3.2. Recodings and Digit Set Conversions

We will see in the remaining of the text that being an high level quantity, the fraction range can be easily tracked through an algorithm and it is therefore very useful when more than one recoding is performed. We have built the three recoder segments presented Figs. 5 to 7. To obtain the corresponding recoder, the desired segment is replicated $m-l+1$ times, to treat all the digits of the input from weight 2^l to weight 2^m as presented later in Fig. 11. Modified versions of the segment are used at the ends, to avoid using logic simply to generate constants. The format and the fraction range are indicated after each recoding.

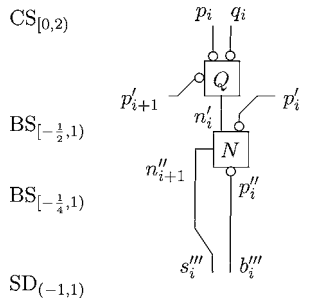


Figure 5. Segment for the translated *NQ* transformation.

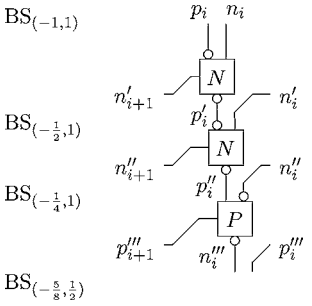


Figure 6. Segment for the PN^2 transformation.

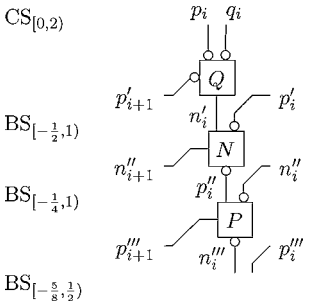


Figure 7. Segment for the *PNQ* transformation.

The first precoder Fig. 5 allows transformation of carry save to sign digit SD. The reader is invited to refer to our [7] report for a more detailed discussion on the signed digit notation. The second and the third precoder of Figs. 6 and 7 convert from borrow save or carry save to radix 4 notation on the digit set $\{-2, \ldots, 2\}$. Our redundant Booth recoder (Section 3.3) uses some additional (lower level) information on the last recoding of these conversions.

3.3. Rounding and Leading Digit Deletion

If we truncate the low order part of a word, we may cause an error up to ± 1 ulp or 2 ulps depending on the notation. On the other hand, if we first apply a PN^{k-1} transformation ($k \geq 1$) to the borrow save register or a $PN^{k-2}Q$ transformation ($k \geq 2$) to a carry save register, we obtain a number stored in borrow save format that has a very limited fraction range included in $[-1/2, 1/2 + 2^{-k})$. As a consequence, truncating it at any position will result in an error less than $1/2 + 2^{-k}$ ulp.

Three lines of half adder provides a PN^2 rounding with fraction range $(-5/8, 1/2)$ sufficient to reduce truncation error below $5/8$ ulps. This can be quite useful in microcoded redundant binary designs for division, square root, and transcendentals.

The next observation has great applicability in extracting differences from a function table for performing interpolation [21, 22]. A $2 \times (m - l + 1)$ bit array may be formed as the difference $a - b$ of two $(m - l + 1)$-bit unsigned integers and stored in borrow save format with no operation. In this case, two recodings allow to discard the leading digits.

Partial compression realizes virtually all the benefits of leading digit deletion. Let X be a positive number represented in borrow save such that $X \leq 2^k - 1$. We know that if $N(X) = [d'_{m+1}\ d'_m \ldots d'_l]$, then $d'_k = 0$ for all $i \geq k + 1$. Thus $N(X)$ may be truncated to a $2 \times (k + 1)$ bit array. If we can only bound the magnitude of X (i.e. $|X| \leq 2^k - 1$), we have to compute $PN(X)$ and we prove that $d''_i = 0$ for all $i \geq k + 2$.

3.4. Booth Recoder

Previous research has shown the feasibility of multiplier designs employing redundant binary operands. To avoid the general increase in hardware size entailed by redundant binary inputs, recent attention has been focused on limiting redundant inputs simply to the multiplier recoder input [23–25] rather than the multiplicand [26, 27].

With minimum circuitry, we are ready to derive the low-power hot-one (only one signal is set at any time) signal controls $\{-2, -1, 0, 1, 2\}$ [24, 28] or the common multiplier controls {negative; doubled factor; unchanged factor} [29, 30]. But we prefer to compute the enhanced sign select controls {negative; positive; doubled factor; unchanged factor} published by Goto et al. [19] who have proved better than previous solutions [31, 32].

On practical grounds, computing directly the control signals is more desirable than converting the

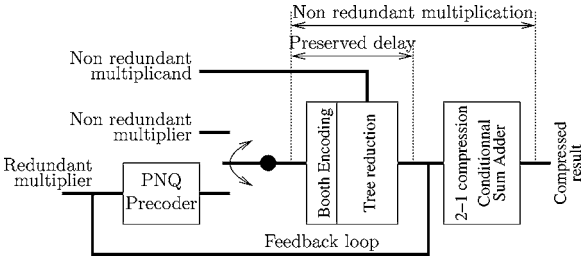


Figure 8. General purpose Booth multiplier with fast feedback capacities through a precoder.

number from digit set $\{0, \ldots, 3\}$ (non redundant input), $\{0, \ldots, 6\}$ (carry save input) or $\{-3, \ldots, 3\}$ (borrow save input) to digit set $\{-2, \ldots, 2\}$ before converting each of the digits obtained to control signals. Past recoders have added critical path delay for the more frequent case where a non redundant binary input is available. Our proposed circuit does not lengthen the time of one multiplication, compared to the state-of-the-art encoding if both inputs are non redundant.

The common radix 4 Booth multiplier computes in three steps the representation of $D = A \times B$ where the two numbers A and B are represented radix 2 by $(a_m \ldots a_l)_2$ and $(b_{m'} \ldots b_{l'})_2$. The final organization of our redundant aware multiplier is visible Fig. 8. The first step converts B to a radix-4 minimally redundant recoding on the digit set $\{-2, \ldots, 2\}$ ("Booth encoding"). This conversion is performed for $\lfloor l'/2 \rfloor \leq j \leq \lceil m'/2 \rceil$ by looking at bits b_{2j+1}, b_{2j} and b_{2j-1} to compute digit b'_j as presented in Table 3. One can check by induction that this operation does not change the represented value as written in the first part of Eq. (19). The second step ("Tree reduction") accumulates the partial products $b'_j \cdot A \cdot 4^j$ in a redundant format to compute the product as suggested by the second part of Eq. (19). The third step converts the redundant result to the usual non redundant binary representation and possibly rounds it according to the active rounding mode ("2-1 compression").

$$\sum_{i=l}^{m} b_i 2^i = \sum_{j=\lfloor l/2 \rfloor}^{\lceil m/2 \rceil} b'_j 4^j \quad \text{and} \quad A \times B$$
$$= \sum_{i=l}^{m} b_i \cdot A \cdot 2^i = \sum_{j=\lfloor l/2 \rfloor}^{\lceil m/2 \rceil} b'_j \cdot A \cdot 4^j \quad (19)$$

Compared to the usual multiplication, Booth recoding divides by two the number of partial products generated and accumulated but these products are more difficult to generate since they cannot be obtained by a `logical and` gate as this is the case when we multiply only by $b_i \in \{0; 1\}$. A naive implementation can ruin all the advantages of Booth recoding. State of the art implementations usually present two cells:

- The encoder is responsible of generating a set of control signals from the input bits b_{2j+1}, b_{2j} and b_{2j-1}. We present in Table 3 its truth table and in the Fig. 9 its high level CMOS circuit.
- The multiplexer computes a representation of $b'_j \cdot A$ based on $(a_m \ldots a_l)_2$ and the control signals generated by the encoder. The number obtained after multiplexing is the one's complement representation of $A \times b'_j$. Little extra circuitry is added to take care of two's complement logic.

As noted by Goto et al. [19], an IEEE-754 standard double precision multiplier uses 27 encoders and 1527

Table 3. Radix 4 Booth recoding.

Multiplier representation			Booth digit b'_j $-2b_{2j+1}$ $+b_{2j}+b_{2j-1}$	Enhanced sign select			
b_{2j+1}	b_{2j}	b_{2j-1}		X_j	TX_j	PL_j	M_j
0	0	0	0	0	1	0	0
0	0	1	1	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	2	0	1	1	0
1	0	0	−2	0	1	0	1
1	0	1	−1	1	0	0	1
1	1	0	−1	1	0	0	1
1	1	1	0	0	1	0	0

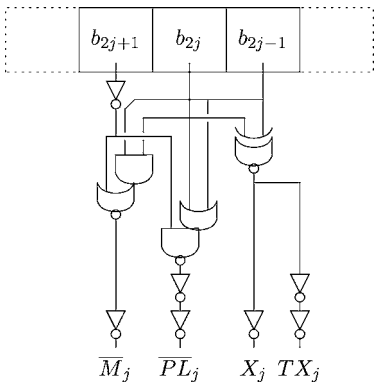


Figure 9. Usual non redundant enhanced sign select Booth encoder.

multiplexers. These later cells may add up to 90% of the area of the circuit.

The next result supports the factoring of multiplier recoding into a two step process. Partial compression is first applied to recode a redundant format so in a second step it may be passed through our modified Booth encoder of Fig. 9 with very few penalty.

The leading negative weight bit of any $2 \times k$ *bit window of a borrow save number* $PN^k(X)$ *indicates the sign of the digit value of that window whenever that digit is non-zero. The same result applies for a window on a* $PN^{k-1}Q$ *recoded number.*

Proof: Let $D = N^k(X)$ with $D = [d_m \dots d_l]$. We define the carry vector $C = [c_{m+1}\, c_m \dots c_{l+1}]$ from Eq. (5) and the result vector $E = P(D) = [e_{m+1}\ e_m \dots e_l]$ from Eq. (6) with $e_i = p_i - n_i$. We obtain the following equation for any combined digit e'_i radix 2^k

$$e'_i = \left\| 0 \cdot e_{(i+1)\cdot k-1} \dots e_l \right\|_2 \cdot 2^k - \| 0 \cdot e_{i\cdot k} \dots e_l \|_2.$$

We extract a term from the radix polynomial of E from Eq. (1)

$$\begin{aligned} e'_i = {} & \left(e_{(i+1)\cdot k-1} - c_{(i+1)\cdot k-1}\right) \cdot 2^{k-1} \\ & + \left\| 0 \cdot c_{(i+1)\cdot k-1}\ e_{(i+1)\cdot k-2} \dots e_l \right\|_2 \cdot 2^k \\ & - \| 0 \cdot e_{k\cdot n} \dots e_l \|_2. \end{aligned}$$

We recognize that $[c_{(i+1)\cdot k-1}\ e_{(i+1)\cdot k-2} \dots e_l]$ is the recoded vector of $[d_{(i+1)\cdot k-2} \dots d_l]$ and $e_{(i+1)\cdot k-1} - c_{(i+1)\cdot k-1} = -n_{(i+1)\cdot k-1}$ and therefore

$$\begin{aligned} e'_i \in {} & -n_{(i+1)\cdot k-1} \cdot 2^{k-1} + 2^{k-1} \cdot (-2^{-k}; 1) \\ & - \frac{1}{2} \cdot (-1 - 2^{-k}; 1) \end{aligned}$$

For any k, if $n_{(i+1)\cdot k-1} = 1$ then $e'_i \le -2^{k-1} + 2^{k-1} + \frac{1+2^{-k}}{2}$ and $e'_i \le 0$ since $e'_i \in \mathbb{Z}$. □

We can now list all the possible outputs of two digits e_{2j+1} and e_{2j} of a PN^2 or a *PNQ* recoder. As presented Table 4 some outputs are not valid because they violate the fraction range or the result just presented above or because they cannot be obtained from the last P recoding.

We have also listed Table 4 all the valid cases if we store b_{2j+1} in n_{2j+1}, b_{2j} both in p_{2j+1} and n_{2j} and finally b_{2j-1} in p_{2j}. The value of the stored number is unchanged since we virtually compute $2B - B$ as presented in details in [7].

Table 4. Radix 4 PN^2 or *PNQ* recoded borrow save number.

Input bits				Radix 4 digit	Acceptable digit?	
p_{2j+1}	n_{2j+1}	p_{2j}	n_{2j}		Recoded	Non redundant
0	0	0	0	0	Yes	Yes
0	0	0	1	−1	No	No
0	0	1	0	1	Yes	Yes
0	0	1	1	0	Yes	No
0	1	0	0	−2	Yes	Yes
0	1	0	1	−3	No	No
0	1	1	0	−1	Yes	Yes
0	1	1	1	−2	Yes	No
1	0	0	0	2	No	No
1	0	0	1	1	Yes	Yes
1	0	1	0	3	No	No
1	0	1	1	2	Yes	Yes
1	1	0	0	0	No	No
1	1	0	1	−1	Yes	Yes
1	1	1	0	1	No	No
1	1	1	1	0	Yes	Yes

We design the new encoder presented Fig. 10 to produce the correct result for each acceptable input of d_{2j+1} and d_{2j}. It shows how a borrow save register can be used as the input to the Booth encoding logic to store either a *PNQ* precoded carry save result obtain through the logic described Fig. 11 or a non redundant number which has been converted to borrow save using

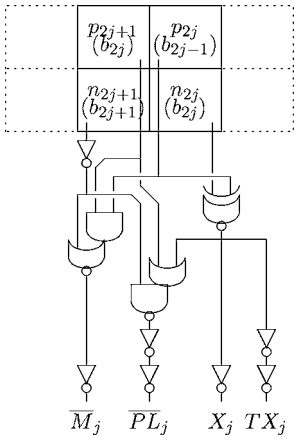


Figure 10. Redundant aware enhanced sign select Booth encoder.

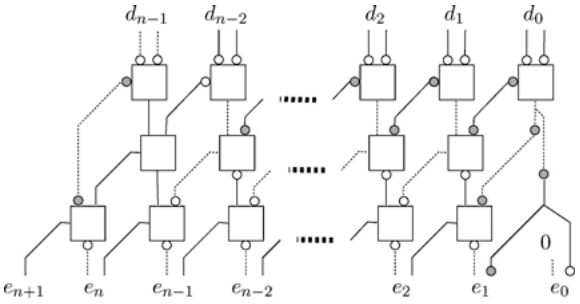


Figure 11. *PNQ* precoder: Convert from carry save to radix 4 Booth recoding.

the centered conversion $2B - B$. The modified Booth multiplier of Fig. 8 accepts a redundant number as one of its operands, by the use of a precoder which does not generate any additional delay for a non-redundant operand.

4. Conclusion

We have presented a general formalism for the study of partial compressions and roundings. This approach proves fruitful in reducing the redundancy of a borrow save or a carry save bit array to allow radix 2^k Booth recoding with minimal circuitry. Reducing redundancy is useful in any application that does not allow any full range redundant number as input but does not require non redundant inputs.

With the technology used in the design presented by Goto et al. [19], that is 0.25 μm CMOS technology with 2.5 V power supply. The authors obtain a 54×54 multiplier with a clock cycle of 4.1 ns. From spice simulations presented by Goto et al. [19] we can predict that, each row of modified half-adders will deliver its output with a delay between 100 ps and 200 ps (depending on electrical properties). As a result, our new circuit can perform a non redundant multiplication in 4.1 ns or prepare a redundant result to be reused by another multiplication in 3.4 ns. For example computing a product of 11 numbers will take 41 ns with the original multiplier and 35 ns with our modified one. Higher savings could be assessed by an electrical simulation of the *PNQ* recoding.

Precoded Booth multipliers have also been implemented [11, 33]. They use 0.5 μm CMOS technology with the Alliance standard cell library [34]. Although this implementation is not as sharp as Goto et al. [19]'s one, Dumonteix [33] have been able to derive interesting results. For example, a 32 bits multiplier with one input recoded producing a redundant result uses 0.81 mm^2 instead of 0.72 mm^2 (12% overhead) for the usual Booth multiplier but the latency drops from 16.83 ns to 12.23 ns (27% improvement).

Other applications arise in the forwarding and feedback of a number internal to a floating point unit to reduce redundancy as presented by Matula and Nielsen [8]. Precoders are used to forward some parts of the redundant result after redundancy is sufficiently reduced. This allows an IEEE standard behavior of the rounding unit albeit the numbers are not compressed to a non redundant format. The construction of a standard adder using this technique is depicted by Nielsen et al. [9].

Other applications will appear in converting the two bounds of an interval to the step of a linear interpolation as it is performed for fast reciprocal and square root approximation [21]. This will allow to forward directly the pair of recoded bounds in borrow save format to a modified Booth multiplier.

Acknowledgments

We wish to thank Peter Kornerup for his notations as presented in [13] and his kind and knowledgeable help over the time of this work.

Notes

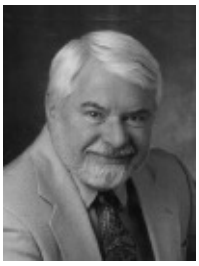
1. The product of a set B by a scalar a is the set of any element of B scaled by a, $a \cdot B = \{a \cdot b, b \in B\}$.
2. The sum of two sets A and B is the set of the sum of any two elements from A and B, $A + B = \{a + b, (a; b) \in A \times B\}$.

References

1. A. Avižienis, "Signed Digit Number Representations for Fast Parallel Arithmetic," *IRE Transactions on Electronic Computers*, vol. 10, 1961, pp. 389–400.
2. D.W. Matula, "Radix Arithmetic: Digital Algorithms for Computer Architecture," in *Applied Computation Theory: Analysis, Design, Modeling*, Prentice Hall, 1976, pp. 374–448.
3. D.W. Matula, "Basic Digit Sets for Radix Representation," *Journal of the ACM*, vol. 29, no. 4, 1982, pp. 1131–1143.
4. A.M. Nielsen and P. Kornerup, "Redundant Radix Representations of Rings," *IEEE Transactions on Computers*, vol. 48, no. 11, 1999, pp. 1153–1165.
5. A.D. Booth, "A Signed Binary Multiplication Technique," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, no. 2, 1951, pp. 236–240.

6. I. Koren, *Computer Arithmetic Algorithms*, Englewood Cliffs, NJ: Prentice Hall, 1993.
7. M. Daumas and D.W. Matula, "Recoders for Partial Compression and Rounding," Research Report 97-01, Laboratoire de l'Informatique du Parallélisme, Lyon, France, 1997.
8. D.W. Matula and A.M. Nielsen, "Pipelined Packet-Forwarding Floating Point: I. Foundations and a Founder," in *Proceedings of the 13th Symposium on Computer Arithmetic*, Monterey, California, T. Lang, J.-M. Muller, and N. Takagi (Eds.), 1997, pp. 140–147.
9. A.M. Nielsen, D.W. Matula, C.N. Lyu, and G. Even, "An IEEE Compliant Floating Point Adder that Conforms with the Pipelined Packet Forwarding Paradigm," *IEEE Transactions on Computers*, vol. 49, no. 4, 2000, pp. 33–47.
10. P.M. Seidel and G. Even, "How Many Logic Levels Does Floating-Point Addition Require?," in *1998 International Conference on Computer Design*, Austin, Texas, 1998, pp. 142–149.
11. Y. Dumonteix and H. Mehrez, "A Family of Redundant Multipliers Dedicated to Fast Computation for Signal Processing," in *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, 2000, pp. 325–328.
12. P.M. Seidel, "High Speed Redundant Reciprocal Approximation," in *3rd Real Numbers and Computers Conference*, Paris, France, 1998, pp. 219–229.
13. P. Kornerup, "Digit-Set Conversion: Generalizations and Applications," *IEEE Transactions on Computers*, vol. 43, no. 5, 1994, pp. 622–629.
14. T.M. Carter and J.E. Robertson, "The Set Theory of Arithmetic Decomposition," *IEEE Transactions on Computers*, vol. 39, no. 8, 1990, pp. 993–1005.
15. M. Ercegovac and T. Lang, "On Recoding in Arithmetic Algorithms," *Journal of VLSI Signal Processing*, vol. 14, 1996, pp. 283–294.
16. M. Daumas and D.W. Matula, "A Booth Multiplier Accepting Both a Redundant or a Non-Redundant Input with no Additional Delay," in *IEEE International Conference on Application-specific Systems, Architectures and Processors*, Boston, Massachusetts, E.E. Swartzlander, G.A. Jullien, and M. Schulte (Eds.), 2000, pp. 205–214.
17. H.P. Sharangpani and M.L. Barton, "Statistical Analysis of Floating Point Flaw in Pentium Processors," White Paper 11, Intel Corporation, 1994.
18. J.-M. Muller, "Algorithmes de division Pour Microprocesseurs: illustration à l'aide du "bug" du Pentium," *Technique et Science Informatiques*, vol. 14, no. 8, 1995.
19. G. Goto et al., "A 4.1ns Compact 54 × 54b Multiplier Utilizing Sign Select Booth Encoders," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, 1997, pp. 1676–1682.
20. A.M. Shams and M.A. Bayoumi, "A Novel High-Performance CMOS 1 Bit Full Adder Cell," *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 47, no. 5, 2000, pp. 478–481.
21. D. Das Sarma and D.W. Matula, "Faithful Bipartite ROM Reciprocal Tables," in *Proceedings of the 12th Symposium on Computer Arithmetic*, Bath, England, S. Knowles and W.H. McAllister (Eds.), 1995, pp. 17–28.
22. M.J. Schulte and J.E. Stine, "Approximating Elementary Functions with Symmetric Bipartite Tables," *IEEE Transactions on Computers*, vol. 48, no. 8, 1999, pp. 842–847.
23. N. Takagi, "Arithmetic Unit Based on a High Speed Multiplier with a Redundant Binary Addition Tree," in *Advanced Signal Processing Algorithms, Architectures and Implementation II*, vol. 1566 of Proceedings of SPIE, 1991, pp. 244–251.
24. B.W.Y. Wei, H. Du, and H. Chen, "A Complex Number Multiplier Using Radix 4 Digits," in *Proceedings of the 12th Symposium on Computer Arithmetic*, Bath, England, S. Knowles and W.H. McAllister (Eds.), 1995, pp. 84–90.
25. C.N. Lyu and D.W. Matula, "Redundant Binary Booth Recoding," in *Proceedings of the 12th Symposium on Computer Arithmetic*, Bath, England, S. Knowles and W.H. McAllister (Eds.), 1995, pp. 50–57.
26. W.S. Briggs and D.W. Matula, "Rectangular Array Signed Digit Multiplier," US Patent 5 184 318, US Patent Office, 1993.
27. W.S. Briggs and D.W. Matula, "A 17 × 69 Bit Multiply and Add Unit with Redundant Binary Feedback and Single Cycle Latency," in *Proceedings of the 11th Symposium on Computer Arithmetic*, Windsor, Ontario, E. Swartzlander, M.J. Irwin, and G. Jullien (Eds.), 1993, pp. 163–170.
28. R.M. Jessani and M. Putrino, "Comparison of Single and Dual Pass Multiply Add Fused Floating Point Units," *IEEE Transactions on Computers*, vol. 47, no. 9, 1998, pp. 927–937.
29. G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 × 54b Regularly Structured Tree Multiplier," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 9, 1992, pp. 1229–1236.
30. N. Ohkubo et al., "A 4.4 ns CMOS 54 × 54-b Multiplier Using Pass Transistor Multiplexer," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, 1995, pp. 251–257.
31. J. Mori et al., "A 10 ns 54 × 54b Parallel Structured Full Array Multiplier with 0.5 μm CMOS Technology," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 4, 1991, pp. 600–605.
32. H. Makino et al., "An 8.8 ns 54 × 54-bit Multiplier with High Speed Redundant Binary Architecture," *IEEE Journal on Solid State Circuits*, vol. 31, no. 6, 1996, pp. 773–783.
33. Y. Dumonteix, "Optimisations des chemins de données arithmétiques par l'utilisation de plusieurs systèmes de numération," Ph.D. Thesis, Université Pierre et Marie Curie, Paris, France, 2001.
34. A. Greiner, L. Lucas, and F. Wajsbürt, "Designing a High Complexity Microprocessor Using the Alliance CAD System," in *Proceedings of the 7th Annual IEEE International ASIC Conference and Exhibit*, Rochester, New York, 1994, pp. 223–226.
35. P. Chai et al., "A 120 MFLOPS CMOS Floating Point Processor," in *Proceedings of the 1991 Custom Integrated Circuits Conference*, San Diego, California, IEEE Computer Society Press, 1991, pp. 15.1.1–15.1.4.
36. H.M. Darley et al., "Floating Point/Integer Processor with Divide and Square Root Functions," US Patent 4 878 190, US Patent Office, 1989.
37. H. Kabuo et al., "Accurate Rounding Scheme for the Newton Raphson Method Using Redundant Binary Representation," *IEEE Transactions on Computers*, vol. 43, no. 1, 1994, pp. 43–51.
38. S.M. Quek, L. Hu, J.P. Prabhu, and F.A. Ware, "Apparatus for Determining Booth Recoder Input Control Signals," US Patent 5 280 439, US Patent Office, 1994.

Marc Daumas is a senior scientist with the French National Center for Scientific Research (CNRS). He has joined the CNRS in 1998 in the *Laboratoire de l'Informatique du Parallélisme* (LIP) in Lyon. He has previously been an assistant professor in the ENSERG in Grenoble (1997), a lecturer (1995–1997) and a fellow student (1989–1993) in the *Ecole Normale Supérieure* (ENS) in Lyon. Marc Daumas received his Ph.D. in 1996 from the ENS in Lyon, and his M.Sc. in 1992 from the Southern Methodist University in Dallas, Texas. He has been working on formal and hardware aspects of computer arithmetic with a special interest in number representation systems.
Marc.Daumas@ENS-Lyon.Fr



David W. Matula received the Ph.D. degree in engineering from the University of California, Berkeley, in 1966. He is currently a professor in the Computer Science and Engineering Department at Southern Methodist University, Dallas, Texas. He is the author of more than 90 papers on computer arithmetic and graph algorithms and holds 13 patents on computer arithmetic and cellular communication systems.
Matula@Engr.SMU.Edu